

## Reconocimiento de objetos en fotografías

Dr. Wilfrido Gómez Flores

### 1 Introducción

El reconocimiento de objetos es la tarea de *encontrar e identificar* automáticamente objetos en una imagen. Los humanos detectamos y reconocemos objetos en imágenes con extrema facilidad, inclusive si los objetos sufren variaciones de forma, tamaño, localización, color, textura, brillo o están parcialmente obstruidos. Por ejemplo, en la Figura 1 se muestran varias imágenes en las cuales identificamos ‘vacas’ sin ninguna duda. Entonces, ¿cómo sabemos que son vacas y no otro objeto? ¿Qué ‘rasgos’ las identifican y las hacen únicas?



Figura 1: Imágenes con variaciones de una vaca.

Aunque para un humano la tarea de reconocimiento de objetos suele ser trivial, es un verdadero desafío para los sistemas de visión por computadora. Generalmente, un sistema de reconocimiento de objetos está diseñado para encontrar e identificar objetos específicos en una imagen, de manera que actualmente no existe un sistema artificial “reconoce-todo”. Sin embargo, una gran variedad de problemas originados en la industria, seguridad, medicina,

milicia, robótica, aeroespacial, etc., se han resuelto mediante la combinación de diversas técnicas de procesamiento de imágenes y reconocimiento de patrones.

Cuando se aborda un problema de reconocimiento de objetos comúnmente se sigue un proceso básico de cinco pasos ilustrado en la Figura 2 [1]:

1. *Adquisición de la imagen*: capturar la escena del mundo real a través de sensores y digitalizarla para su procesamiento, almacenamiento y transmisión.
2. *Preprocesamiento*: aplicar técnicas de mejoramiento de contraste, reducción de ruido, realce de características, etc., de modo que la imagen se adecúe para los siguientes pasos.
3. *Segmentación*: aislar los objetos de interés de la imagen.
4. *Extracción de rasgos*: describir numéricamente la naturaleza de los objetos segmentados como su forma, color, textura, etc.
5. *Clasificación*: asignar una clase o categoría a cada objeto de la imagen basado en sus rasgos.

La *base de conocimiento* incluye el conocimiento previo del problema que será resuelto, de modo que sirva en el diseño de cada etapa para la selección de las técnicas de procesamiento de imágenes y reconocimiento de patrones más adecuadas.

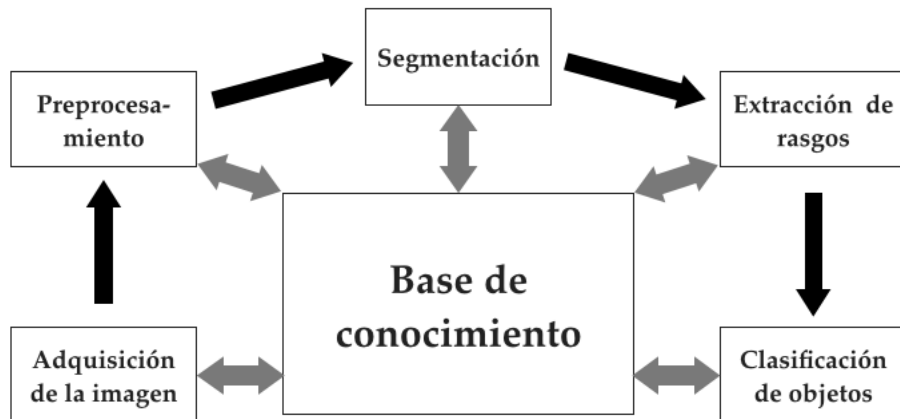


Figura 2: Procesos básicos en el reconocimiento de objetos.

En este taller se abordará un problema común dentro de un ambiente industrial, en el cual se deben clasificar automáticamente una serie de objetos que pasan por una banda de producción para posteriormente separarlos y empacarlos. Para resolver este problema se implementarán las etapas del diagrama mostrado en la Figura 2. Se revisarán las bases teóricas de cada técnica que será utilizada en el proceso de reconocimiento y se hará la implementación en el lenguaje de programación MATLAB. Al finalizar el taller, el alumno

tendrá una panorámica general del diseño de un sistema de reconocimiento de objetos de propósito específico. A continuación se describirá el desarrollo de las diferentes etapas del sistema de reconocimiento de objetos en la Figura 2.

## 2 Base de conocimiento

El problema fundamental es reconocer automáticamente cinco tipos de piezas de ferretería que pasan por una banda de producción, las cuales se muestran en la Figura 3.

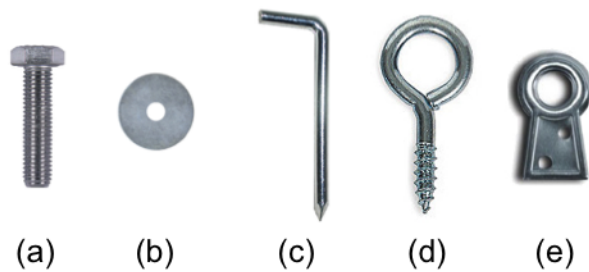


Figura 3: Piezas de ferretería: (a) tornillo, (b) rondana, (c) alcayata, (d) armella, (e) grapa cola de pato.

El sistema de adquisición de imágenes es una cámara convencional que proporciona fotografías de tamaño  $320 \times 240$  píxeles y 8-bits de profundidad, es decir, 256 niveles de gris. En las fotografías pueden haber un número indeterminado de piezas diferentes y repetidas e inclusive puede no aparecer algún tipo de pieza en específico como se ilustra en la Figura 4.

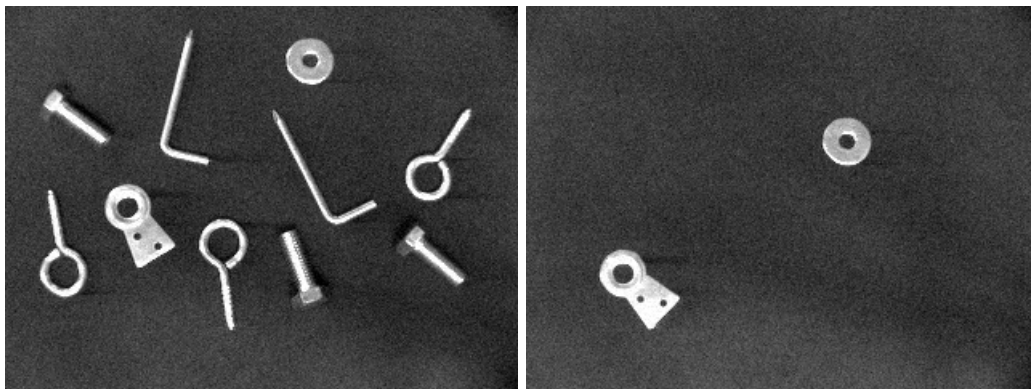


Figura 4: Fotografías típicas obtenidas con el sistema de adquisición de imágenes.

Al observar detenidamente las imágenes obtenidas se pueden determinar las siguientes características:

- Las imágenes están contaminadas con un poco de ruido.
- La iluminación de la escena no es constante para todo el plano, de modo que existen zonas más oscuras que otras.
- Los objetos brillantes están relativamente bien contrastados en relación al fondo oscuro.
- Los cinco tipos de piezas poseen formas distintas.
- Los objetos pueden estar en diferentes posiciones, es decir, existe rotación y traslación en los objetos.

Considerando el conocimiento previo sobre las características de las imágenes, se propone un sistema de reconocimiento de piezas de ferretería que consta de los siguientes objetivos:

1. Preprocesamiento: reducir el ruido y corregir la iluminación.
2. Segmentación: generar una imagen binaria con los objetos separados de el fondo.
3. Extracción de rasgos: medir cuantitativamente la forma de los objetos segmentados mediante medidas invariantes a traslaciones y rotaciones.
4. Clasificación: reconocer los objetos mediante la búsqueda de la pieza de referencia más parecida.

## 3 Preprocesamiento

### 3.1 Reducción de ruido

#### 3.1.1 Teoría

Una definición simple de ruido es cualquier componente indeseada que deteriora la calidad de la imagen. El ruido es inherente en imágenes, se genera principalmente durante el proceso de adquisición y tiene un comportamiento aleatorio.

Cuando se realiza filtrado de ruido en imágenes, hay que tomar en cuenta el compromiso entre la reducción del ruido y la deformación de la imagen. Esto quiere decir, mientras más se trate de reducir el ruido, más difuminación de la imagen se produce, lo cual puede distorcionar la forma original de los objetos. Este efecto indeseado sucede generalmente en filtros lineales como el filtro Gaussiano [1]. En la Figura 5 se observa que al aumentar tanto el valor de la desviación estándar como el tamaño del filtro Gaussiano se obtiene una mejor reducción de ruido aunque el objeto se difumina en mayor medida.

Para reducir eficientemente el ruido y preservar la forma original de los objetos se suele utilizar el filtro no lineal de difusión anisotrópico (ADF) [2]. Este tipo de filtro reduce iterativamente el ruido dentro de las regiones delimitadas por los bordes de los objetos y no lo hace a través de ellos. Para lograr esto se requiere detectar los contornos de los objetos

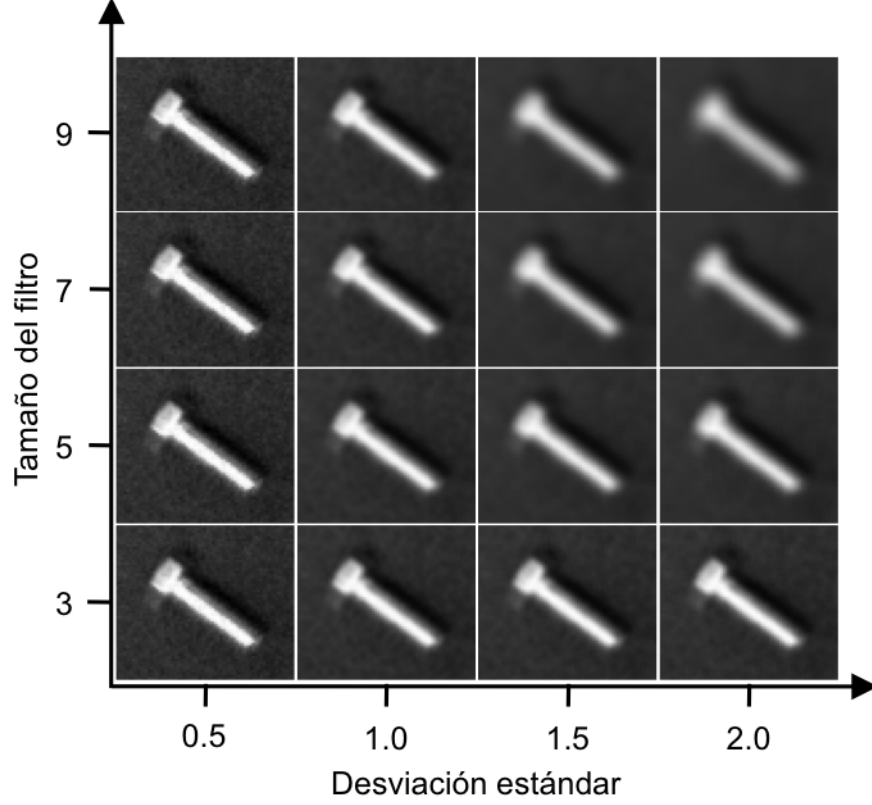


Figura 5: Filtrado Gaussiano con diferentes valores de desviación estándar y tamaño de filtro.

para indicarle al filtro que los píxeles del borde deben mantenerse prácticamente intactos. Entonces, el proceso de filtrado ADF se expresa como:

$$I_{i,j}^{t+1} = I_{i,j}^t + \frac{1}{4} [g(\nabla_N I) \cdot \nabla_N I + g(\nabla_S I) \cdot \nabla_S I + g(\nabla_W I) \cdot \nabla_W I + g(\nabla_E I) \cdot \nabla_E I]_{i,j}^t, \quad (1)$$

donde  $I_{i,j}^t$  es un píxel con coordenadas  $(i, j)$  en la iteración  $t$ , el operador  $\nabla$  detecta el borde del objeto en las direcciones  $N$  (arriba),  $S$  (abajo),  $E$  (derecha) y  $W$  (izquierda) mediante las siguientes diferencias:

$$\begin{aligned} \nabla_N I &= I_{i-1,j} - I_{i,j}, \\ \nabla_S I &= I_{i+1,j} - I_{i,j}, \\ \nabla_E I &= I_{i,j+1} - I_{i,j}, \\ \nabla_W I &= I_{i,j-1} - I_{i,j}. \end{aligned} \quad (2)$$

Nótese que el detector de bordes efectúa la resta entre el píxel  $I_{i,j}$  y cuatro vecinos alrededor de él. Por tanto, cuando el píxel  $I_{i,j}$  se encuentra en una región homogénea (i.e., intensidades similares) el valor de la diferencia será pequeña (tiende a cero). De manera contraria, si el píxel  $I_{i,j}$  se encuentra sobre un borde (i.e., intensidades diferentes) el valor de la diferencia será grande (tiende a infinito).

La función de difusión  $g(\cdot)$  en la Ecuación 1 indica el nivel de filtrado que debe aplicarse al píxel  $(i, j)$ , el cual se calcula como:

$$g(\nabla I) = \frac{1}{1 + (|\nabla I|/\kappa)^2}, \quad (3)$$

donde  $|\cdot|$  indica el valor absoluto y  $\kappa$  es una constante que controla la extensión del filtrado. Por tanto, cuando  $|\nabla I| \rightarrow \infty$  entonces  $g \rightarrow 0$  y cuando  $|\nabla I| \rightarrow 0$  entonces  $g \rightarrow 1$ .

En la Figura 6 se ilustra el proceso de filtrado ADF para una sola iteración. Nótese que en las imágenes correspondientes al coeficiente de conducción  $g$ , los píxeles claros poseen valores cercanos a '1', lo cual favorece la reducción de ruido en esas regiones, mientras que los píxeles oscuros tienen valores cercanos a '0', lo cual cancela el efecto del filtrado.

$$I^{t+1} = I^t + \frac{1}{4} \times \left( g_N \nabla I_N + g_S \nabla I_S + g_W \nabla I_W + g_E \nabla I_E \right)$$

Figura 6: Ejemplo de filtrado ADF para una iteración.

### 3.1.2 Implementación

La implementación en MATLAB del filtro de difusión anisotrópico incluye dos funciones: `adf`, programa principal que resuelve iterativamente la Ecuación 1, y `grads`, detector de bordes en la Ecuación 2. El código de ambas funciones se muestran en las Figuras 7 y 8, respectivamente.

En la Figura 9 se muestra un ejemplo del filtrado ADF aplicado a una imagen ruidosa. Nótese que la cantidad de ruido se redujo considerablemente, mientras que los bordes del objeto se preservaron.

```

function J = adf(J)
iter = 100;      % Numero maximo de iteraciones
lambda = 0.25;  % Parametro de estabilidad
kappa = 3;      % Constante de difusion
J = double(J);  % Convierte la imagen a tipo de dato double
[Mi,Ni] = size(J); % Tamano de la imagen MxN
E = [1, 1:Mi-1]; % Indices de los pixeles a la derecha
W = [2:Mi, Mi];  % Indices de los pixeles a la izquierda
S = [1, 1:Ni-1]; % Indices de los pixeles hacia abajo
N = [2:Ni, Ni];  % Indices de los pixeles hacia arriba
for t = 1:iter % Iteraciones
    [DN,DS,DE,DW] = grads(J,N,S,E,W); % Detector de bordes (Ecuacion 2)
    % Coeficientes de difusión (Ecuacion 3)
    gN = 1./(1+((abs(DN)/kappa).^2)); % Hacia arriba
    gS = 1./(1+((abs(DS)/kappa).^2)); % Hacia abajo
    gE = 1./(1+((abs(DE)/kappa).^2)); % Hacia la derecha
    gW = 1./(1+((abs(DW)/kappa).^2)); % Hacia la izquierda
    % Actualiza la imagen para la siguiente iteracion
    J = J + lambda*(gN.*DN + gS.*DS + gE.*DE + gW.*DW); % (Ecuacion 1)
end
J = uint8(J); % Imagen filtrada convertida a 8-bits

```

Figura 7: Código del filtro de de difusión anisotrópico.

```

function [DN,DS,DE,DW] = grads(I,N,S,E,W)
DN = I(:,N) - I; % Diferencia hacia arriba
DS = I(:,S) - I; % Diferencia hacia abajo
DE = I(E,:) - I; % Diferencia hacia la derecha
DW = I(W,:) - I; % Diferencia hacia la izquierda

```

Figura 8: Código del detector de bordes.

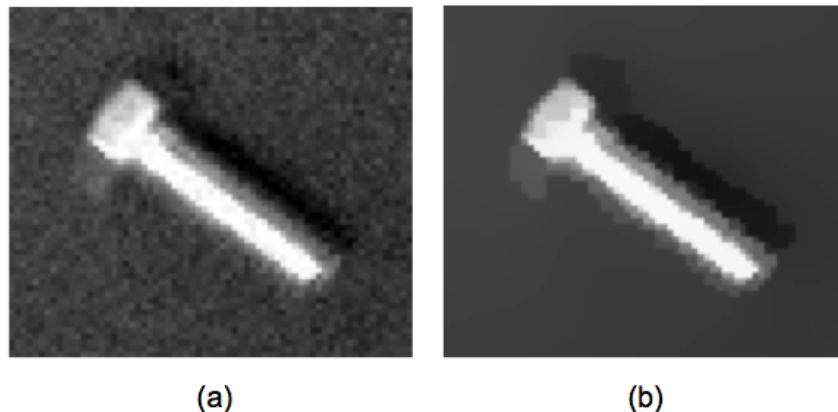


Figura 9: (a) Imagen original con ruido. (b) Imagen filtrada mediante ADF.

## 3.2 Corrección de iluminación

### 3.2.1 Teoría

La iluminación en la imagen es generada por la posición de la fuente luminosa (e.g., una lámpara) en relación a la lente de la cámara. Una única fuente luminosa fija produce mayor claridad en los objetos más cercanos y se oscurecen gradualmente a medida que se alejan. Este fenómeno se refleja como una variación del brillo en diferentes zonas de la imagen y puede afectar el desempeño del método de segmentación utilizado en la siguiente etapa.

Para resolver este problema se debe homogeneizar la iluminación y un método efectivo es el filtrado Top-Hat. Este filtro realiza la resta de la imagen original,  $I$ , menos la imagen resultante después de someter  $I$  a un proceso de erosión y dilatación con un elemento estructurante (o ventana)  $w$  mayor al tamaño de los objetos en la imagen [3]. El filtro Top-Hat se expresa como:

$$\text{TH}_w(I) = I - \delta_w[\varepsilon_w(I)], \quad (4)$$

donde  $\delta$  y  $\varepsilon$  son los operadores de dilatación y erosión, respectivamente, y se computan como:

$$\delta_w(I) = \max_{(s,t) \in w} \{I(x-s, y-t)\}, \quad (5)$$

$$\varepsilon_w(I) = \min_{(s,t) \in w} \{I(x-s, y-t)\}. \quad (6)$$

Los operadores  $\delta$  y  $\varepsilon$  indican que se toma el valor de intensidad máximo y mínimo, respectivamente, de una ventana  $w$  y se coloca en el píxel central como se ilustra en la Figura 10.

### 3.2.2 Implementación

El filtro Top-Hat primero realiza una erosión, para borrar todos los objetos de la imagen, seguido de una dilatación, para recuperar las intensidades del fondo, y el resultado se resta a la imagen original, para cancelar el efecto de el fondo.

Este filtro se puede programar fácilmente utilizando las funciones `imerode` e `imdilate` contenidas en la *Image Processing Toolbox* de MATLAB. En la Figura 11 se muestra el código que implementa el filtro Top-Hat en la Ecuación 4.

En la Figura 12 se ilustra el proceso de filtrado Top-Hat para una imagen filtrada previamente por ADF. Nótese que el fondo se homogeneiza contrastando mejor los objetos.

## 4 Segmentación

### 4.1 Teoría

Una vez que la imagen ha sido preprocesada, los objetos deben separarse automáticamente de el fondo mediante un umbral de intensidad,  $t$ , de modo que la imagen segmentada es



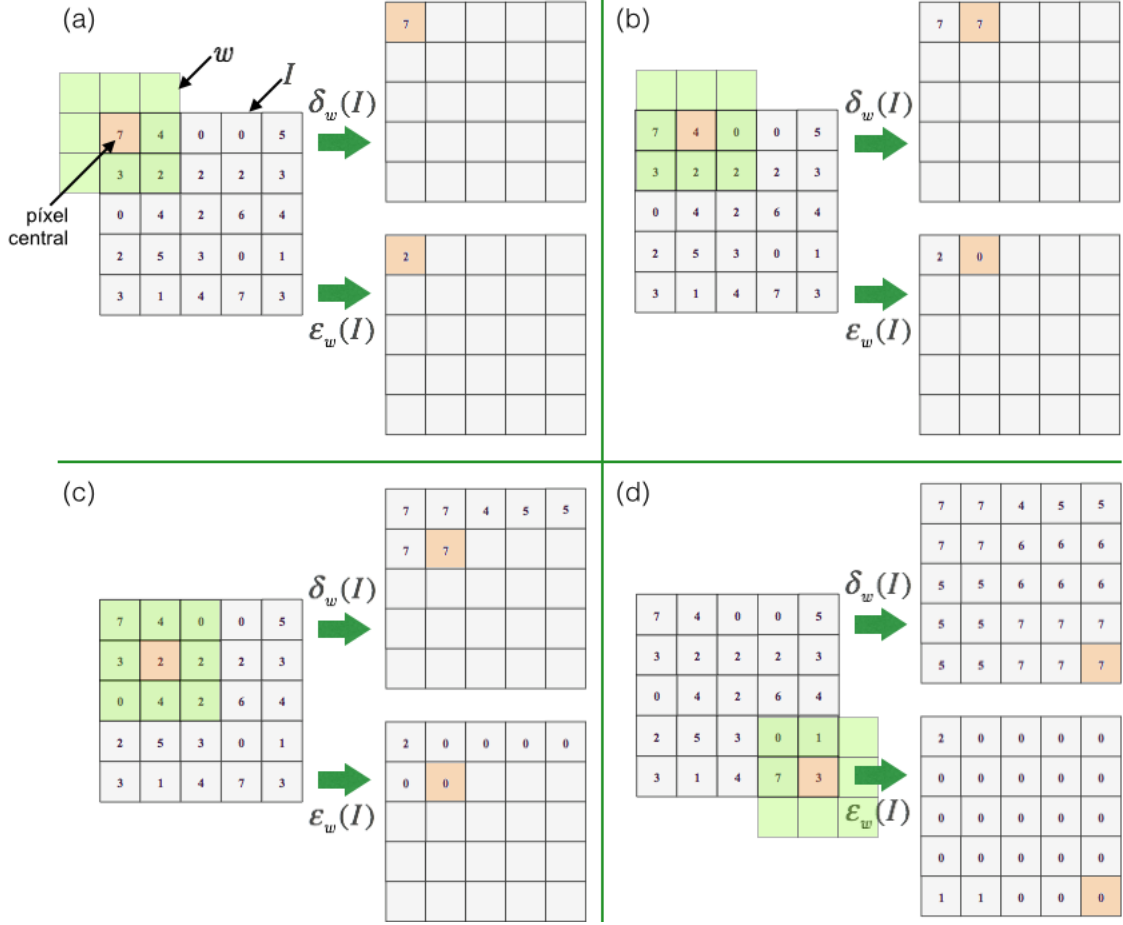


Figura 10: Procesos de dilatación ( $\delta$ ) y erosión ( $\varepsilon$ ) de una imagen ( $I$ ) de  $5 \times 5$  píxeles utilizando una ventana ( $w$ ) de  $3 \times 3$  píxeles. Nótese el desplazamiento de  $w$  sobre cada píxel de  $I$ ; se ilustran los resultados parciales cuando se toman los valores máximos y mínimos de los vecindarios alrededor de los píxeles: (a)  $I(1,1)$ , (b)  $I(1,2)$ , (c)  $I(2,2)$  y (d)  $I(5,5)$ .

una imagen binaria,  $B$ , donde el valor '0' corresponde a un píxel del fondo y el valor '1' corresponde a un píxel del objeto, lo cual se expresa como:

$$B_{i,j} = \begin{cases} 1, & \text{si } I_{i,j} \geq t \\ 0, & \text{otro caso} \end{cases}. \quad (7)$$

Un método de umbralado de imágenes ampliamente utilizado en la práctica es el método de Otsu [4], el cual analiza el histograma de niveles de gris de la imagen para agrupar los píxeles en las clases fondo y objetos. El histograma de la imagen es un arreglo unidimensional, donde cada elemento contiene el número de píxeles existentes para un nivel de gris

```

function TH = tophat(I,t)
% Elemento estructurante cuadrado de tamaño txt
w = strel('square',t);
% Realiza la erosión (Ecuación 6)
E = imerode(I,w);
% Realiza la dilatación (Ecuación 5)
D = imdilate(E,w);
% Resta la dilatación a la imagen original (Ecuación 4)
TH = imsubtract(I,D);

```

Figura 11: Código del filtro Top-Hat.

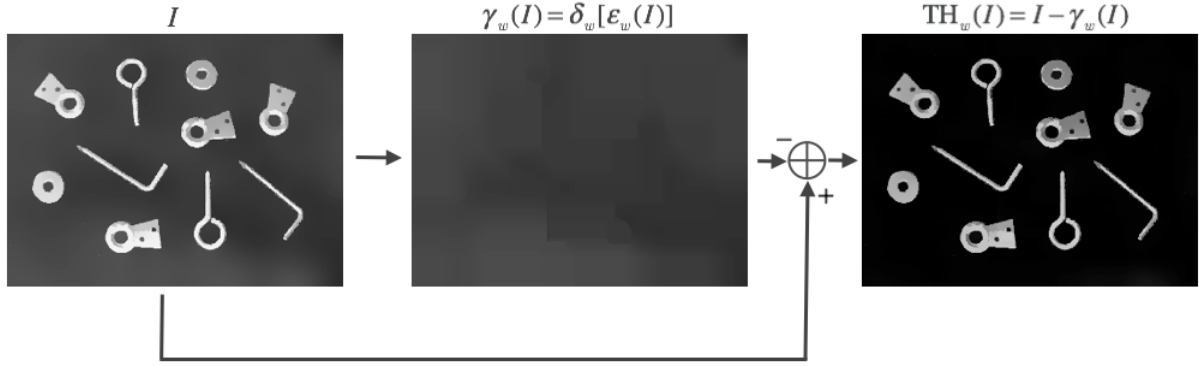


Figura 12: Corrección de iluminación con filtrado Top-Hat.

en específico, como se ilustra en la Figura 13.

Para encontrar el umbral de intensidad óptimo,  $t^*$ , que separa los objetos de el fondo, se hace un barrido secuencial de todos los niveles de gris del histograma en el rango  $[0, 255]$ , de modo que cada nivel de gris  $t = 0, 1, \dots, 255$  es un umbral que separa a los objetos de el fondo generando una imagen binaria mediante la Ecuación 7. El método de Otsu mide la *varianza interclase* que existe para cada nivel de gris  $t$ , es decir, cuantifica la relación de las variaciones de los niveles de gris que están asociados al fondo y a los objetos. Por tanto, el umbral óptimo  $t^*$  se encuentra en el nivel de gris que maximiza la varianza interclase como:

$$t^* = \arg \max_{t=0,1,\dots,255} \left\{ A_t \cdot (A_{255} - A_t) \cdot (\mu_t - \nu_t)^2 \right\}, \quad (8)$$

donde  $\mu_t$  y  $\nu_t$  son los promedios de los niveles gris del fondo y los objetos, respectivamente, para el umbral  $t$ , los cuales se calculan como:

$$\mu_t = \frac{B_t}{A_t} \text{ y } \nu_t = \frac{B_{255} - B_t}{A_{255} - A_t}, \quad (9)$$

donde las sumas acumuladas  $A_t$  y  $B_t$  se computan como:

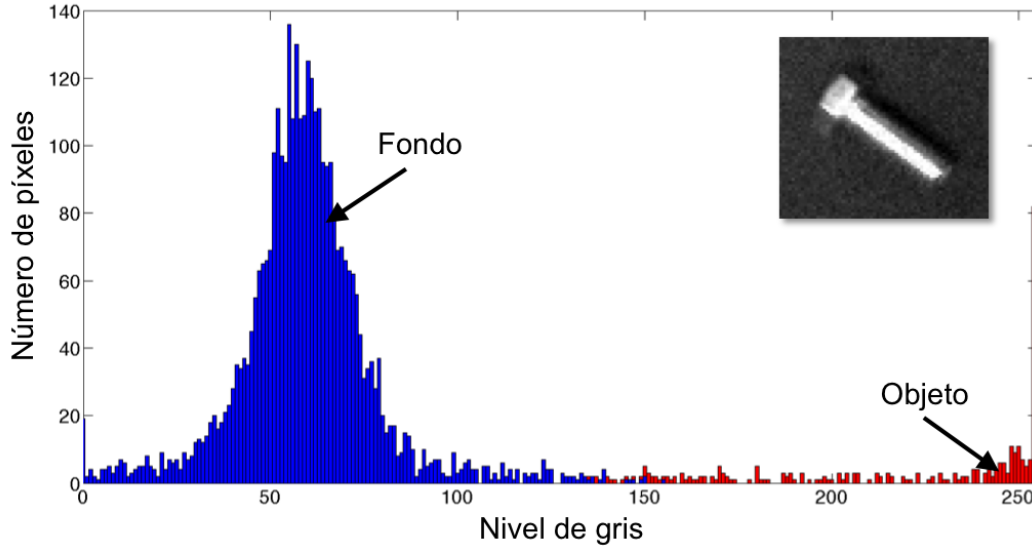


Figura 13: Histograma de niveles de gris para la imagen del tornillo.

$$A_t = \sum_{i=0}^t h_i \text{ y } B_t = \sum_{i=0}^t i \cdot h_i. \quad (10)$$

donde  $h_i$  es el histograma de niveles de gris de la imagen.

En la Figura 14 se ilustra el proceso segmentación con el método de Otsu para una imagen que ha sido previamente preprocesada.

## 4.2 Implementación

El método de Otsu se implementa en MATLAB mediante el código mostrado en la Figura 15. Por otro lado, utilizando las funciones `im2bw` y `graythresh`, contenidas en la *Image Processing Toolbox* de MATLAB, también se puede implementar el método de Otsu. Por ejemplo, si  $I$  contiene la imagen en niveles de gris, entonces la imagen segmentada  $B$  se obtiene mediante la instrucción:  $B = \text{im2bw}(I, \text{graythresh}(I))$ .

En el siguiente proceso de extracción de rasgos es necesario analizar cada objeto de manera individual. Sin embargo, debido a que la salida del método de segmentación es una imagen binaria, donde los píxeles con valor '1' involucran a todos los objetos, el sistema de reconocimiento aún no sabe dónde se localiza cada objeto de manera individual. Por tanto, se debe aplicar un método de etiquetado que transforme la imagen binaria en una imagen donde todos los píxeles que pertenecen a un sólo objeto tengan una etiqueta numérica única  $L = 1, 2, \dots, N$ , donde  $N$  es el número total de objetos en la imagen.

Para implementar el etiquetado se utiliza la función `bwlabel`, contenida en la *Image Processing Toolbox* de MATLAB. Esto se realiza invocando la siguiente instrucción:

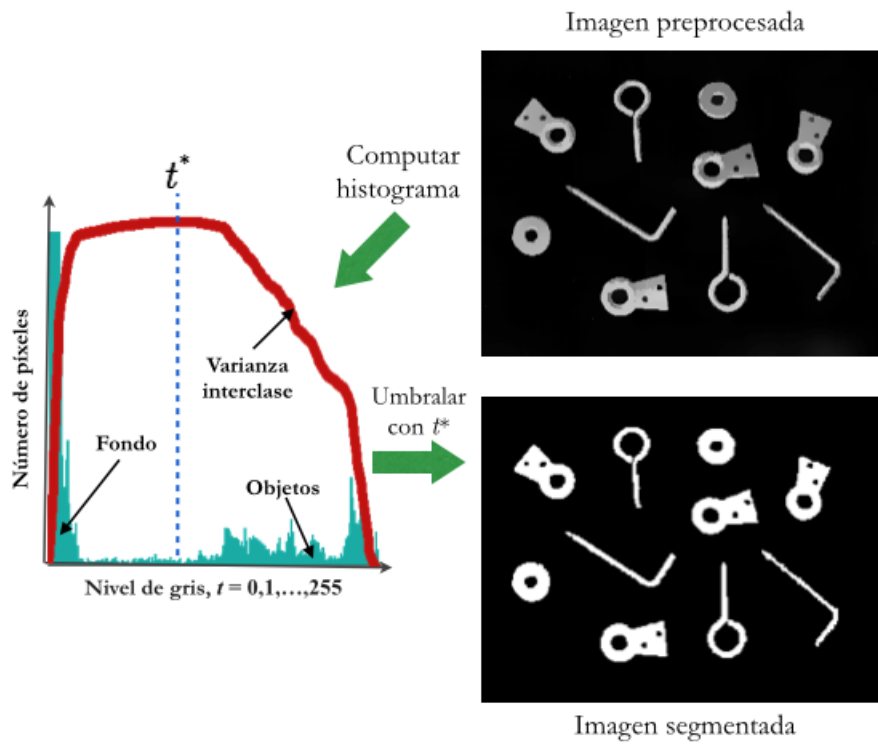


Figura 14: Segmentación con método de Otsu.

```
function B = otsu(I)
% Imagen de 8-bits (256 niveles de gris)
L = 2^8;
% Numero de pixeles en la imagen
N = numel(I(:));
% Computa el histograma de niveles de gris
hi = accumarray(I(:)+1,ones(N,1),[L 1],@sum,0);
% Computa sumas parciales (Ecuacion 10)
At = cumsum(hi)+eps;
Bt = cumsum((0:L-1)'.*hi)+eps;
A255 = At(L); B255 = Bt(L);
% Computa las medias de cada clase (Ecuacion 9)
mu_t = Bt./At;
nu_t = (B255-Bt)./(A255-At)+eps;
% Maximiza la varianza interclase (Ecuacion 8)
[~,t] = max(At.*(A255-At).*(mu_t-nu_t).^2);
% Segmenta la imagen de entrada (Ecuacion 7)
B = I > (t-1);
```

Figura 15: Código del método de Otsu.

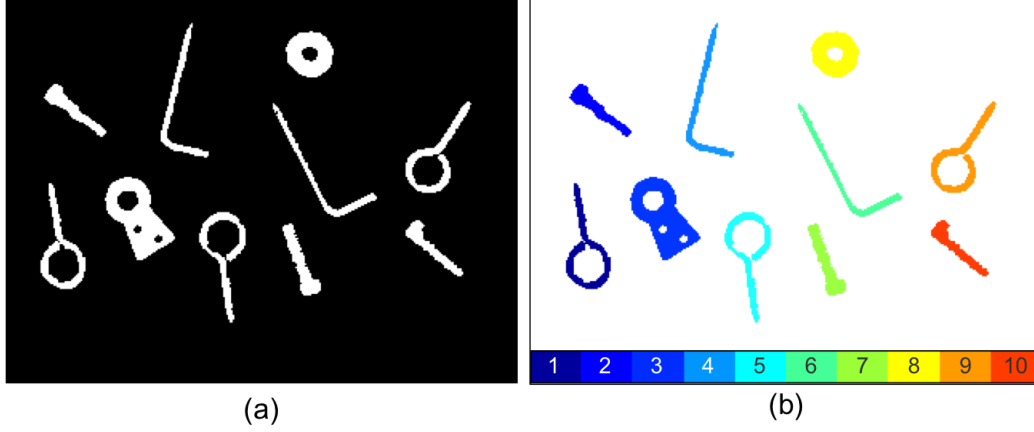


Figura 16: (a) Imagen binaria obtenida del método de Otsu. (b) Imagen etiquetada con 10 objetos.

$L = \text{bwlabeled}(B)$ , donde  $B$  es la imagen segmentada. En la Figura 16 se muestra el resultado del método de etiquetado de una imagen binaria.

## 5 Extracción de rasgos

### 5.1 Teoría

La forma característica de un objeto puede cuantificarse mediante *momentos*, los cuales describen la manera en que se distribuyen los píxeles de un objeto sobre el plano de la imagen. Los momentos deben ser invariantes (i.e., valores similares para objetos del mismo tipo) a las transformaciones geométricas (traslación, rotación y escalamiento) que pueden sufrir los objetos y al mismo tiempo deben ser discriminantes (i.e., valores distintos para objetos de diferente tipo). Estas características son deseables para poder reconocer los objetos con mayor facilidad.

Los momentos de Hu [5] son un conjunto de siete descriptores invariantes que cuantifican la forma de un objeto, donde los dos primeros momentos se computan como:

$$\begin{aligned}\phi_1 &= \eta_{20} + \eta_{02}, \\ \phi_2 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2,\end{aligned}\tag{11}$$

donde los momentos centrales normalizados de segundo orden se calculan como:

$$\begin{aligned}
\eta_{20} &= \frac{m_{20} - \frac{m_{10}^2}{m_{00}}}{m_{00}^2}, \\
\eta_{02} &= \frac{m_{02} - \frac{m_{01}^2}{m_{00}}}{m_{00}^2}, \\
\eta_{11} &= \frac{m_{11} - \frac{m_{10}m_{01}}{m_{00}}}{m_{00}^2},
\end{aligned} \tag{12}$$

y los momentos geométricos de orden  $p + q$  se calculan como:

$$m_{pq} = \sum_x \sum_y x^p y^q I(x, y). \tag{13}$$

donde  $I(x, y)$  es un píxel del objeto con coordenadas  $(x, y)$ .

En la Figura 17 se muestran los valores numéricos de los momentos de Hu para tres tipos de piezas de ferretería con diferentes rotaciones. Nótese que los valores  $\phi_1$  y  $\phi_2$  son muy similares para objetos del mismo tipo y diferentes entre objetos distintos.

## 5.2 Implementación

El cálculo de los dos primeros momentos de Hu involucra dos funciones: `hu_moments`, programa principal que computa los momentos  $\phi_1$  y  $\phi_2$  en la Ecuación 11, y `momgeom`, calcula los momentos geométricos de orden  $p + q$  en la Ecuación 13. El código de ambas funciones se muestran en las Figuras 18 y 19, respectivamente.

# 6 Clasificación de objetos

## 6.1 Teoría

En la extracción de rasgos, un objeto es caracterizado mediante un vector que contiene  $d$  características,  $\mathbf{x} = [x_1, x_2, \dots, x_d]$ , denominado patrón, el cual está asociado a su etiqueta de clase  $\mathbf{y} \in \{1, 2, \dots, C\}$ , la cual indica la pertenencia a un tipo de objeto en específico entre  $C$  diferentes clases de objetos. Nótese que para el problema de reconocimiento de piezas de ferretería un objeto se representa por un vector de dos características,  $\mathbf{x} = [\phi_1, \phi_2]$ , y está asociado a una de las cinco clases conocidas ( $C = 5$ ): tornillo ( $\mathbf{y} = 1$ ), rondana ( $\mathbf{y} = 2$ ), alcayata ( $\mathbf{y} = 3$ ), armella ( $\mathbf{y} = 4$ ) y grapa cola de pato ( $\mathbf{y} = 5$ ).

El reconocimiento de objetos se puede realizar mediante la comparación entre el objeto que se desea reconocer y muestras de referencia de cada tipo de objeto conocido. Así el objeto se asigna al grupo de muestras de referencia que más se parezca. A las muestras de referencia se les denomina *patrones de entrenamiento*, los cuales involucran muestras representativas de las distintas clases de objetos con diferentes variaciones como rotaciones, traslaciones y cambios de escala. Por tanto, el conjunto de entrenamiento captura un conocimiento previo de las posibles variaciones que un objeto puede presentar.
















Tornillo					
					
$\phi_1 = [$	0.54	0.50	0.45	0.47	0.52 $]$
$\phi_2 = [$	0.25	0.21	0.17	0.19	0.24 $]$
<hr/>					
Rondana					
					
$\phi_1 = [$	0.19	0.19	0.19	0.19	0.19 $]$
$\phi_2 = [$	0.10	0.40	0.25	0.33	0.25 $]$
<hr/>					
Armella					
					
$\phi_1 = [$	0.79	0.78	0.77	0.83	0.71 $]$
$\phi_2 = [$	0.37	0.36	0.35	0.41	0.27 $]$

Figura 17: Valores de los momentos de Hu  $\phi_1$  y  $\phi_2$  para tres objetos diferentes con rotaciones.

El reconocimiento de objetos usando el ‘más parecido’ se puede relacionar con la idea de ‘cercanía’ en el sentido de distancia. Entonces, un objeto cercano a otro indica que son parecidos y su similitud se reduce a medida que se alejan. Un método que realiza la comparación entre objetos utilizando la distancia Euclidiana es el algoritmo de los  $k$ -vecinos más cercanos (KNN, *k-nearest neighbor*) [6]. El término ‘ $k$ ’ denota el número de muestras de entrenamiento más cercanas contra las cuales se va comparar un patrón que será clasificado. De este modo, la idea fundamental del KNN se puede enunciar como sigue: ‘Si camina como pato, hace como pato, y parece como pato, entonces es probable que sea un pato’. Trasladando la definición de distancia Euclidiana al problema de reconocimiento de piezas de ferretería, para un par de objetos  $\mathbf{x}_i = [\phi_1^i, \phi_2^i]$  y  $\mathbf{x}_j = [\phi_1^j, \phi_2^j]$ , se tiene:

$$D(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\phi_1^i - \phi_1^j)^2 + (\phi_2^i - \phi_2^j)^2}. \quad (14)$$

El algoritmo KNN requiere la ejecución de los siguientes pasos:

1. Medir la distancia Euclidiana entre el patrón que se desea reconocer y todas las muestras del conjunto de entrenamiento.

```

function hu = hu_moments(I)
% Convierte imagen a double
I = double(I);
% Tamaño MxN de la imagen
[N,M] = size(I);
% Crea plano coordenado (x,y)
[x,y] = meshgrid(0:M-1,0:N-1);
% Calcula momentos geométricos (Ecuacion 13)
m00 = momgeom(I,x,y,0,0);
m10 = momgeom(I,x,y,1,0);
m01 = momgeom(I,x,y,0,1);
m20 = momgeom(I,x,y,2,0);
m02 = momgeom(I,x,y,0,2);
m11 = momgeom(I,x,y,1,1);
% Calcula momentos centrales normalizados (Ecuacion 12)
n20 = (m20-((m10^2)/m00))/m00^2;
n02 = (m02-((m01^2)/m00))/m00^2;
n11 = (m11-((m10*m01)/m00))/m00^2;
% Calcula momentos de Hu 1 y 2 (Ecuacion 11)
phi1 = n20 + n02;
phi2 = (n20-n02)^2 + 4*n11^2;
hu = [phi1,phi2];

```

Figura 18: Código de los momentos de Hu  $\phi_1$  y  $\phi_2$ .

```

function mpq = momgeom(I,x,y,p,q)
mpq = sum(sum(I.*(x.^p).*(y.^q))); % (Ecuacion 13)

```

Figura 19: Código de los momentos geométricos de orden  $p + q$ .

2. Identificar los  $k$  patrones de entrenamiento más cercanos al patrón que se desea reconocer (i.e., aquellos que obtuvieron la menor distancia Euclidiana) y obtener la etiqueta de clase de cada uno.
3. El patrón que se desea reconocer se asigna a la clase que obtuvo el mayor número de ocurrencias o votos.

En la Figura 20 se ilustra la clasificación de un patrón desconocido  $\mathbf{x}$  a la clase que obtuvo el mayor número de votos de entre sus cinco vecinos más cercanos.

## 6.2 Implementación

Si se cuenta con la *Statistics and Machine Learning Toolbox* de MATLAB, la búsqueda de los  $k$  vecinos más cercanos se puede realizar con la función `knnsearch`. En la Figura 21 se muestra el código de la función `knn1` que implementa el algoritmo KNN para  $k = 3$ .

En caso que no se tenga disponible la función `knnsearch`, el algoritmo KNN se puede implementar utilizando la función alternativa `knn2` mostrada en la Figura 22.



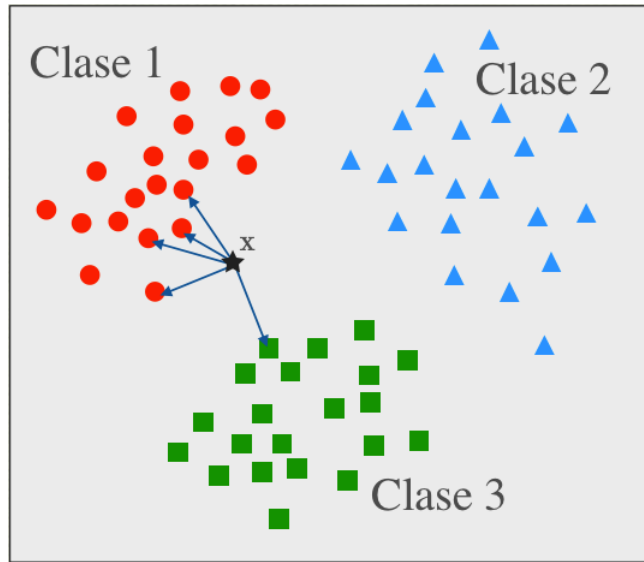


Figura 20: El patrón  $\star$  se compara con sus  $k = 5$  vecinos más cercanos y se asigna a la Clase 1, ya que obtuvo el mayor número de votos, esto es 4 de 5.

```
function Yrec = knn1(Xent,Yent,Xrec)
% Numero de muestras u objetos que seran reconocidos
NumObj = size(Xrec,1);
% Busca los indices de los 3 vecinos de entrenamiento mas cercanos
idx = knnsearch(Xent,Xrec,'dist','euclidean','k',3);
% Identifica la clase de los 3 vecinos mas cercanos
C = Yent(idx);
% Inicializa en ceros la matriz de conteo de votos
V = zeros(NumObj,max(Yent));
for i = 1:max(Yent) % Para cada clase conocida
    V(:,i) = sum(C==i,2); % Cuenta el numero de veces que aparece la clase
end
% La clase con el mayor numero de votos es la ganadora
[~,Yrec] = max(V,[],2);
```

Figura 21: Código del algoritmo KNN que utiliza la función `knnsearch` para la búsqueda de los  $k$  vecinos más cercanos.

En la Figura 23 se muestra el *espacio de características* generado por los patrones de entrenamiento de las piezas de ferretería. Cada muestra representa la coordenada dada por el vector  $\mathbf{x} = [\phi_1, \phi_2]$ . En cada clase hay 20 muestras con variaciones de rotación y traslación. Nótese que se generaron cinco grupos separados y compactos, lo cual indica que los rasgos son discriminantes e invariantes, de modo que el algoritmo KNN particiona adecuadamente el espacio de características utilizando  $k = 3$  vecinos más cercanos.

```

function Ypred = knn2(Xent,Yent,Xrec)
% Numero de muestras u objetos que seran reconocidos
NumObj = size(Xrec,1);
% Calcula la distancia Euclidiana entre los patrones de entrenamiento
% y los patrones que se desean reconocer
D = sqrt(bsxfun(@plus,dot(Xrec',Xrec',1),dot(Xent',Xent',1))- 2*(Xent*Xrec'));
% Ordena las distancias de cada muestra de menor a mayor
[~,nearest] = sort(D,1);
% Extrae los indices de los 3 vecinos de entrenamiento mas cercanos
idx = nearest(1:3,:);
% Identifica la clase de los vecinos mas cercanos
C = Yent(idx);
% Inicializa en ceros la matriz de conteo de votos
V = zeros(NumObj,max(Yent));
for i = 1:max(Yent) % Para cada clase conocida
    V(:,i) = sum(C==i,2); % Cuenta el numero de veces que aparece la clase
end
% La clase con el mayor numero de votos es la ganadora
[~,Ypred] = max(V,[],2);

```

Figura 22: Código del algoritmo KNN.

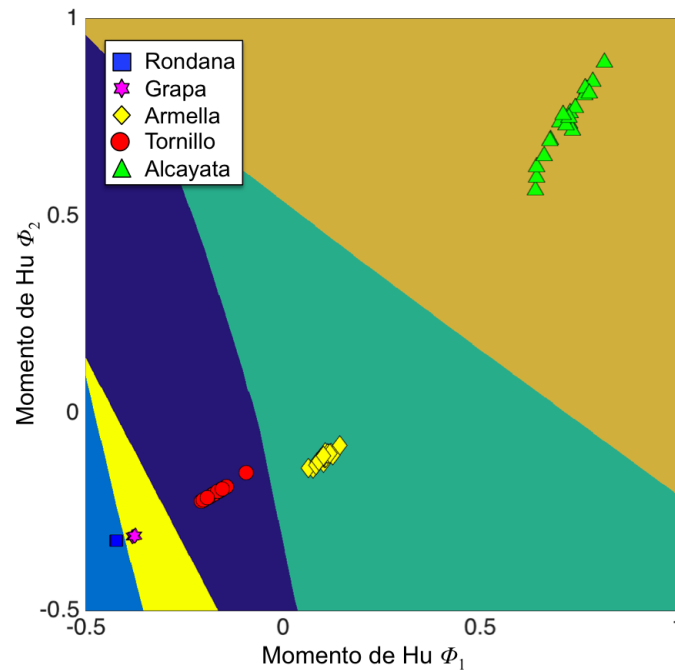


Figura 23: Espacio de características generado por los momentos de Hu  $\phi_1$  y  $\phi_2$  para las piezas de ferretería y su partición mediante el algoritmo KNN con  $k = 3$  vecinos más cercanos.

## 7 Implementación del sistema de reconocimiento de objetos

El sistema de reconocimiento de objetos incluye la conjunción en secuencia de todas las funciones descritas anteriormente: filtrado ADF  $\rightarrow$  filtrado Top-Hat  $\rightarrow$  método de Otsu  $\rightarrow$  etiquetado  $\rightarrow$  extracción de momentos de Hu  $\rightarrow$  algoritmo KNN. En la Figura 24 se muestra el código que realiza el reconocimiento de piezas de ferretería. Nótese que el archivo `entrenamiento.mat` contiene almacenados los patrones de entrenamiento, `Xent` (normalizados en el rango  $[-1, +1]$  mediante la función `softmaxnorm`), y su etiqueta de clase, `Yent` (valores enteros de 1 a 5). También, en el archivo `entrenamiento.mat` están almacenadas la media, `MED`, y desviación estándar, `DE`, de los patrones de entrenamiento, las cuales son utilizadas para normalizar en el rango  $[-1, +1]$  los patrones que se van a clasificar.

```
load('entrenamiento.mat');
% ***** PREPROCESAMIENTO *****
% Reduce el ruido con Filtro de difusion anisotropico
I1 = adf(I0);
% Correccion de iluminacion con Top-Hat
I2 = tophat(I1,75);
% ***** SEGMENTACION *****
% Umbralado con metodo de Otsu
I3 = otsu(I2);
% Elimina regiones pequenas aisladas menores a 20 pixeles de area
I4 = bwareaopen(I3,20);
% ***** EXTRACCION DE CARACTERISTICAS *****
labels = bwlabel(I4); % Etiquetado de la imagen
NumObj = max(labels(:)); % Numero de objetos
Xrec = zeros(NumObj,2); % Inicializa vector de características
for i = 1:NumObj % Para cada objeto
    BW = labels==i; % Identifica el objeto con la etiqueta i
    I5 = crop_roi(BW); % Recorta la region de interes
    Xrec(i,:) = hu_moments(I5); % Calcula los momentos de Hu
end
% ***** CLASIFICACION *****
Xrec = softmaxnorm(Xrec,[MED;DE]); % Normaliza los datos
Yrec = knn1(Xent,Yent,Xrec); % Algoritmo KNN, Yrec es la clase asignada a
% cada objeto en Xrec
```

Figura 24: Implementación del sistema de reconocimiento de objetos.

Finalmente, en la Figura 25 se observa que el sistema de reconocimiento detectó y clasificó correctamente las ocho piezas de ferretería en la imagen.

## 8 Agradecimiento

Las imágenes utilizadas en este taller fueron proporcionadas por el Dr. Juan Humberto Sossa Azuela, profesor investigador del Centro de Investigación en Computación del I.P.N.

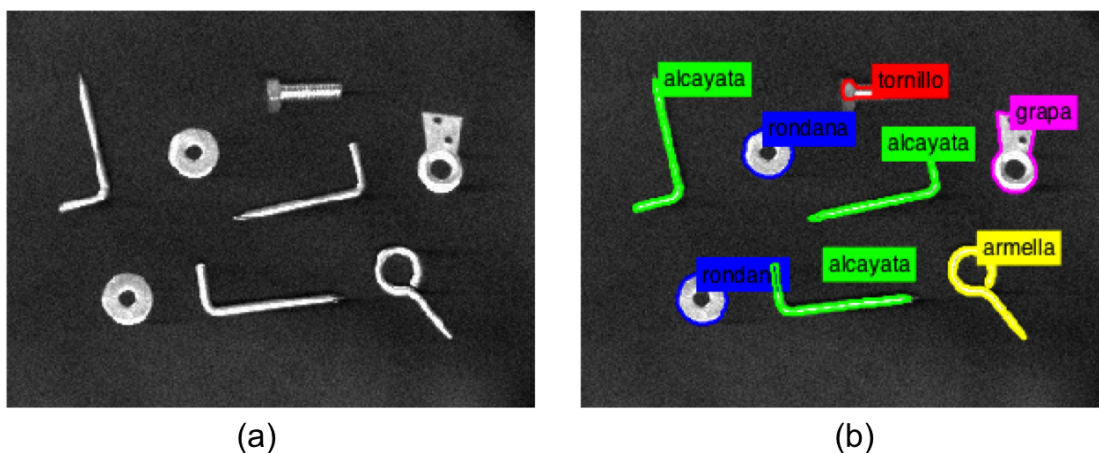


Figura 25: (a) Fotografía con piezas de ferretería. (b) Detección y clasificación correcta de ocho objetos en la imagen.

## Referencias

- [1] Rafael C. Gonzalez, Richard E. Woods, *Digital Image Processing*, Pearson Prentice Hall, New Jersey, USA, 3rd edition, 2008.
- [2] P. Perona, J. Malik, *Scale-space and edge detection using anisotropic diffusion*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 12(7): 629-639, 1990.
- [3] P. Soille, *Morphological Image Analysis*, Springer-Verlag, Berlin, Germany, 2nd edition, 2004.
- [4] N. Otsu, *A threshold selection method from gray-level histogram*, IEEE Transactions on Systems, Man, and Cybernetics, 9(1): 62-66, 1979.
- [5] M. K. Hu, *Visual pattern recognition by moment invariants*, IRE Transactions on Information Theory, 8(2): 179-187, 1962.
- [6] S. Theodoridis, K. Koutroumbas, *Pattern Recognition*, Academic Press, Canada, 4th edition, 2009.